

# Interaction and observation: categorical semantics of reactive systems through dialgebras

Vincenzo Ciancia

*Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche (IT)*  
*Institute for Logic, Language and Computation, University of Amsterdam (NL)*

**Abstract.** We use dialgebras, generalising both algebras and coalgebras, as a simple categorical model aimed to describe the semantics of an interactive system by the means of reaction rules. The focus is on providing a model, alternative to coalgebras, where interaction is built-in, instead of relying on a (possibly difficult) understanding of the side effects of a component in isolation. Kernel equivalence in dialgebras can be used as a standard notion of behavioural equivalence, determined by how a given process interacts with the others, and the obtained observations. We develop a technique to compare equivalences of different categories of dialgebras. This includes comparing a given dialgebraic semantics of a calculus to an existing coalgebraic semantics. We use the Calculus of Communicating Systems to exemplify usage of the framework.

## 1 Introduction

A system is called *interactive* when its semantics depends upon interaction with a surrounding environment. The semantics does not just yield a value (or not at all), but rather it consists in the denotation of the *behaviour* of the system itself, usually described either by *reaction rules* or by a *labelled transition system* (LTS). The difference is illustrated by the following example, defining a reaction rule for the synchronisation of two parallel processes in a process calculus (the rule on the left) or the LTS variant (the three rules on the right):

$$\frac{\text{true}}{a.P \parallel \bar{a}.Q \rightarrow P \parallel Q} \quad \frac{\text{true}}{a.P \xrightarrow{a} P} \quad \frac{\text{true}}{\bar{a}.P \xrightarrow{\bar{a}} P} \quad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\bar{a}} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'}$$

Here  $a.P$  is a process waiting for a signal on channel  $a$ , whose continuation is  $P$ . Similarly,  $\bar{a}.P$  sends a signal on  $a$ , while  $P \parallel Q$  is the parallel composition of two processes. The reaction rule may be read as “whenever two processes can synchronise, they do, and evolve into the parallel composition of their continuations”. The LTS rules may be read as: “whenever a process can send or receive a signal, it evolves into its continuation, and has a *side effect* on the environment”. Then two processes with complementing side effects are allowed to interact by the last rule. Notice that it may appear conceptually odd at first that an input is “performed” as an action by a process, when the idea behind input is that a process *waits* for it. The labels in an LTS may be considered as the minimal

amount of information needed to predict the behaviour of a process when inserted into *any* context; the input label makes no exception, so it “predicts” that a signal may be received, and the process can continue.

LTSs are widely used for modelling the semantics of interactive systems, since they come equipped with bisimilarity, a form of *behavioural equivalence* that specifies when the semantics of two processes is the same. Coalgebras generalise LTSs and have a standard definition of bisimilarity, coinciding with kernel equivalence of morphisms under mild assumptions. Many formalisms received a coalgebraic treatment (e.g. the  $\pi$ -calculus and calculi with name fusions). This becomes increasingly harder as the complexity of the calculus grows, depending on the general question of *what are side effects* in a specific calculus (e.g. name allocation in the  $\pi$ -calculus). The issue is avoided when using reduction rules.

Once established that coalgebras model side-effecting computation, one may seek for a more direct approach, where reaction rules are the main object of study, and side effects or labels are not needed at all to define behavioural equivalence. In this work, we do this using a generalisation of coalgebras, namely *dialgebras* (introduced in §3), to represent a rule system as an object in a category, so that kernel equivalence can be used as a notion of behavioural equivalence.

To appreciate the difference, consider a function  $f : X \rightarrow \mathcal{P}(L \times X)$ . It specifies an LTS with states in  $X$  describing, for each state  $x$ , the (non-deterministic) choices at  $x$ , the side effects of each choice, and the resulting state. In LTSs, and coalgebras in general, elements are observed in isolation. In contrast, a function  $f : X \times X \rightarrow \mathcal{P}(X)$  is a kind of dialgebra, as we shall see. The value of  $f(x, y)$  is meant to describe the possible (non-deterministic) outcomes of an interaction between  $x$  and  $y$ . Elements are not observed in isolation, but rather their mutual interactions define the semantics.

Dialgebras have been first used in computer science for the categorical specification of data types [1,2]. A systematic study of dialgebras, patterned after [3], has been initiated in [4]. In [5], we modelled asynchronous process calculi as isolated machines that can be fed with input tokens by an external observer, using dialgebras to generalise *Mealy machines*. In this paper we aim at a more intensional characterisation, tailored to reaction rules, obtained by studying interaction between pairs of systems rather than the relation between their input and output. More generally, our work diverts from previous research in categorical program semantics, as we take a “local” approach to study dialgebras. That is, instead of studying a whole category of dialgebras, or comparing two such categories, one just considers objects that are reachable by morphisms from the particular dialgebra being studied, or from the dialgebras being compared. This is motivated by the fact that a final dialgebra fails to exist, so there is no universal semantic domain for the whole class of considered objects. Thus, we concentrate on defining and studying the *bisimilarity quotient* of a specific system. We show that such an object exists under mild conditions, even when there is no final object. The bisimilarity quotient is sufficient to provide a canonical semantics to an interactive system up-to behavioural equivalence, by the means of a canonical epimorphism.

To clarify locality, notice that a universal model also serves the purpose to compare elements from different systems of the same type; this ability is lost when switching to dialgebras. However, we may confidently say that this feature is typically not used in program semantics, as different systems generally have different types. Even in the case when two systems *can* be compared in this way, the result is arguably artificial: such a comparison is only sensible if the labels, that is the side effects, have the same intended interpretation in all the considered systems; this is an underlying assumption in coalgebras. A local point of view is that a universal meaning of side effects is not encountered very often in practice: each system has its own means to let its elements interact. Locality also some complexity from applications. For example, conditions on the functors such as boundedness, or weak pullback preservation, are not required.

Milner's CCS is our leading example, both for its simplicity, allowing us to illustrate our theory in a clear way, and since the pre-existing LTS semantics can be compared to the one we define. In §4 we introduce the type of dialgebras for CCS and the associated behavioural equivalence. In §5 we provide a dialgebraic semantics of the calculus. In order to compare it to the LTS semantics, in §6, we develop a proof technique which is based on *bisimulation invariant* functions, that permits one to transform a dialgebra of one type into a dialgebra of another type while preserving bisimulation. In §7 this technique is used to prove the equivalence between the LTS and dialgebraic semantics of CCS. More examples can be easily developed, once the basic theory is established (see Remark 1).

A categorical semantics of reactive systems is also provided by the research line started in [6,7,8]. Roughly, an LTS is derived from the reaction rules and its bisimilarity is used as the chosen behavioural equivalence. One considers all the unary contexts  $C[-]$  of the term algebra, and defines an LTS labelled by contexts, having a transition  $x \xrightarrow{C[-]} y$  whenever  $C[x] \rightarrow y$ . Nevertheless, *minimal* contexts need to be carefully selected in order to obtain a sensible equivalence relation; the obtained semantics depends on this choice, and is not directly specified by the reaction rules themselves. In addition, the resulting categorical framework is highly non-trivial, and it requires several ad-hoc constructions (consider e.g. relative pushouts and related notions); dealing with structural congruence introduces further complications. Dialgebras provide a mathematically simpler theory, and easily account for structural congruence. However, compositionality, which is a fundamental result in the other framework, has not yet been studied. Indeed, compositionality is of great interest. Bringing bialgebraic constructions [9] to dialgebras whose carrier is a term algebra is an important future direction.

Here, we focus on a basic theory where an additional algebraic structure on system states may be useful, but not necessary, to describe the semantics. In a nutshell, we may say that we propose dialgebras as an alternative to *coalgebras*, not to *bialgebras*: (dialgebraic) interaction is intended to play the role of (coalgebraic) side effects in defining behavioural equivalence. Just like coalgebras can be endowed with distributive laws and turned into bialgebras, we expect that similar results will be obtained as an additional layer on top of the basic one.

## 2 The coalgebraic semantics of CCS

The *Calculus of Communicating Systems* (CCS) is a simple process calculus emphasizing just one aspect of computation: *communication* between parallel processes. In the *pure* variant only *synchronisation* is considered, that is, the exchanged data is not taken into account. We briefly recall the LTS (thus, coalgebraic) semantics of CCS here. The interested reader may refer to [10] for more details. The syntax is described by the grammar:

$$P ::= \sum_{i \in I} \alpha_i.P_i \mid P_1 \parallel P_2 \mid (\nu a)P \quad \alpha ::= \tau \mid a \mid \bar{a}$$

where  $I$  is a finite set, and  $a$  ranges over a countable set of channels  $\mathcal{C}$ . Elements of  $P$  are *processes*, or *agents*. Elements of  $\alpha$  are *atomic actions*, or *prefixes*, or *guards*. CCS features operators for denoting: *parallel composition* ( $P_1 \parallel P_2$ ); *restriction* of a channel  $x$  which becomes private to  $P$  ( $(\nu a)P$ ); non-deterministic choice among a finite set of action-prefixed processes ( $\sum_{i \in I} \alpha_i.P_i$ ), usually written as  $a_1.P_1 + \dots + a_n.P_n$ . Special cases of the choice construct are the empty process  $\emptyset$  which is the sum of zero processes, and the action prefix  $\alpha.P$ , which is the sum of one process. The actions  $\alpha$  are: the *internal step* ( $\tau$ ); the act of *receiving* a signal on channel  $a$  (the action  $a$ ); *sending* a signal on a channel ( $\bar{a}$ ). We omit recursion for simplicity; including it does not change the presented results. Channels are also referred to as *names*. Given a process  $P$ , the sets of its *free names*  $fn(P)$  and *bound names*  $bn(P)$  are defined as usual by induction; the only way to introduce bound names is by the name restriction operator, in which  $x$  is bound. In the following, let  $X$  be the set of CCS processes.

Structural congruence is the minimal equivalence relation  $\equiv \subseteq X \times X$  that includes  $\alpha$ -conversion of the bound variable  $a$  in  $(\nu a)P$ ; commutative monoid axioms for the parallel operator with respect to  $\emptyset$ ; the equations  $(\nu a)(P \parallel Q) \equiv ((\nu a)P) \parallel Q$ ,  $(\nu a)\emptyset \equiv \emptyset$ ,  $(\nu a)(\nu b)P \equiv (\nu b)(\nu a)P$  for all  $P, Q$ ,  $a \notin fn(Q)$  and  $b$ .

The labelled transition system for CCS is presented in Figure 1. The set  $L$  of labels is just the set of prefixes  $\alpha$ . We write  $x \xrightarrow{\alpha} y$  as a shorthand for  $(\alpha, y) \in g(x)$ . An LTS with labels from  $L$  can be represented as a pair  $(X, f)$  where  $f$  is a function<sup>1</sup> from  $X$  to  $\mathcal{P}(L \times X)$ . Let  $(X, g)$  be the LTS for CCS.

**Definition 1.** *Bisimilarity of CCS programs is the greatest symmetric relation  $\sim_g \subseteq X \times X$  such that, whenever  $x \sim_g y$  and  $x \xrightarrow{\alpha} x'$ , there is  $y'$  such that  $y \xrightarrow{\alpha} y'$  and  $x' \sim_g y'$ .*

Notice that we are defining bisimilarity in one specific system (e.g. CCS), not between states of different systems. This corresponds to a standard categorical notion once recognised that LTSs are  $\mathcal{P}(L \times -)$ -coalgebras.

<sup>1</sup> Here  $\mathcal{P}(X)$  is the (*co-variant*) *power set* of  $X$ . In coalgebras it is typical to assume a cardinality bound on the size of the subsets, as the functor  $\mathcal{P}$  does not have a final coalgebra. Indeed, in all the systems we define,  $\mathcal{P}$  can be replaced by a bounded variant, as all our transition sets are finite or countable. Since we do not use the final coalgebra in this paper, the distinction is immaterial here.

$$\begin{array}{c}
\frac{true}{\alpha.P + Q \xrightarrow{\alpha} P} (pre) \quad \frac{x \notin fn(\alpha) \quad P \xrightarrow{\alpha} P'}{(\nu a)P \xrightarrow{\alpha} (\nu a)P'} (res) \quad \frac{P \xrightarrow{\alpha} P'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q} (par) \\
\\
\frac{P \xrightarrow{\bar{c}} P' \quad Q \xrightarrow{c} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'} (syn) \quad \frac{P \equiv Q \quad P' \equiv Q' \quad P \xrightarrow{\alpha} P'}{Q \xrightarrow{\alpha} Q'} (str)
\end{array}$$

**Fig. 1.** The LTS describing the operational semantics of CCS.

**Definition 2.** Given a functor  $\mathbf{T} : \mathbf{Set} \rightarrow \mathbf{Set}$ , a  $\mathbf{T}$ -coalgebra is a pair  $(X, f : X \rightarrow \mathbf{T}X)$  where  $X$  is a set. A homomorphism of coalgebras from  $(X, f)$  to  $(Y, g)$  is a function  $h : X \rightarrow Y$  such that  $g \circ h = \mathbf{T}h \circ f$ .  $\mathbf{T}$ -coalgebras and their morphisms form the category  $\mathbf{Coalg}(\mathbf{T})$ .

**Definition 3.** Given a  $\mathbf{T}$ -coalgebra  $(X, f)$ , coalgebraic bisimilarity  $\sim_f \subseteq X \times X$  is defined by  $x \sim_f y \iff \exists (Y, g). \exists h : (X, f) \rightarrow (Y, g). h(x) = h(y)$ .

It is well-known that under suitable conditions on  $\mathbf{T}$  (that the functor for LTSs respects) bisimilarity as in Definition 3 coincides with other coalgebraic notions (see e.g. [11]). We do not discuss the details, but we note that for LTSs and the one for CCS in particular, the relations from Definition 1 and 3 coincide.

### 3 Dialgebras

A coalgebraic semantics can be interpreted as the discerning power of an *observer* that can see all the actions done by a process. With *dialgebras*, we can endow the observer with the ability to *interact* with the system. The passive observer becomes an entity which runs *experiments* and *observes* the results. By this change of point of view, we can represent e.g. input as an experiment in which the observer feeds a system with a value [5], or we can represent interaction as a binary experiment involving two processes as we do in the current paper.

We restrict all our definitions to the category  $\mathbf{Set}$ , but indeed the theory of dialgebras can be developed in any category.

**Definition 4.** Given two functors  $\mathbf{F}, \mathbf{B} : \mathbf{Set} \rightarrow \mathbf{Set}$ , a  $(\mathbf{F}, \mathbf{B})$ -dialgebra is a pair  $(X, f)$  where  $X$  is the carrier set or underlying set and  $f : \mathbf{F}X \rightarrow \mathbf{B}X$  is a function. A  $(\mathbf{F}, \mathbf{B})$ -dialgebra homomorphism from  $(X, f)$  to  $(Y, g)$  is a function  $h : X \rightarrow Y$  such that  $g \circ \mathbf{F}h = \mathbf{B}h \circ f$ , as depicted in Figure 2. Dialgebras and their homomorphisms form the category  $\mathbf{Dialg}(\mathbf{F}, \mathbf{B})$ .

Roughly,  $\mathbf{F}$  is the syntax of experiments, of which the function  $f$  is the semantics, yielding a set of elements in a type of observed results  $\mathbf{B}$ . The crucial feature of dialgebras is that, since they form a category, they have a standard notion of equivalence coming from kernels of morphisms. Noting that the category  $\mathbf{Coalg}(\mathbf{T})$  is the same as  $\mathbf{Dialg}(\mathbf{Id}, \mathbf{T})$ , coalgebraic bisimilarity from Definition 3 becomes an instance of it.

$$\begin{array}{ccccc}
X & & FX & \xrightarrow{f} & BX \\
\downarrow h & & \downarrow Fh & & \downarrow Bh \\
Y & & FY & \xrightarrow{g} & BY
\end{array}$$

**Fig. 2.** A dialgebra homomorphism.

**Definition 5.** Given a  $(F, B)$ -dialgebra  $(X, f)$ , dialgebraic bisimilarity  $\sim_f \subseteq X \times X$  is defined by  $x \sim_f y \iff \exists(Y, g). \exists h : (X, f) \rightarrow (Y, g). h(x) = h(y)$ .

Notice that in Definition 5 we implicitly consider the kernel of  $h$ . This kernel is taken on  $h$  as a function. Thus, we do not require kernels (pullbacks of a function with itself) in  $Dialg(F, B)$ . The “extensional” definition that we provide is applicable to any kind of dialgebra (independently from  $F$  and  $B$ ), and it avoids the machinery of relation liftings which is more complicated for dialgebras (see [2]). We now study epi-mono factorisations of dialgebras.

**Proposition 1.** *If  $F$  and  $B$  preserve monos whose domain is empty, the category  $Dialg(F, B)$  has unique epi-mono factorisations. Otherwise, it has epi-mono factorisations of morphisms whose domain does not have an empty carrier.*

Proposition 1 guarantees that bisimilarity is determined by the epimorphisms. In coalgebras, the kernel of the unique morphism into the final object (if any) coincides with bisimilarity. For simple functors  $F$  such as  $F(X) = X \times X$  even when  $B$  is bounded, a final dialgebra does not exist<sup>2</sup>.

*Example 1.* Let  $F(X) = X \times X$  and  $B(X) = \mathcal{P}_{fin}(X)$  (the finite power set of  $X$ ). Suppose there is a final dialgebra  $(Z, z)$ . Consider the dialgebra  $(Z + 1, f)$  where  $f(x, y) = z(x, y)$  if  $x, y \in Z$ , while  $f(x, *) = f(*, x) = \{*\}$ , and  $f(*, *) = \emptyset$ . Consider the final map  $h : (Z + 1, f) \rightarrow (Z, z)$ . Certainly  $h$  is injective and surjective on  $Z$  (by finality of  $(Z, z)$ ), thus there is  $x \in Z$  such that  $h(x) = h(*)$ . Then  $h$  is not a dialgebra homomorphism: we have  $z(Fh(*, x)) = z(h(*), h(x)) = z(h(*), h(*)) = z(Fh(*, *))$ , while  $Bh(f(*, x)) = \{h(*)\} \neq \emptyset = Bh(f(*, *))$ . The element  $*$  behaves differently from every other element in  $Z$  in one experiment, but the set  $Z$  encompasses all the possible behaviours, which leads to a paradox.

Similarly, for an arbitrary set  $X$ , define the dialgebra  $g(x, x) = \{x\}, g(x, y) = \emptyset$  if  $x \neq y$ . Since  $X$  is arbitrary, and no different elements are bisimilar, the cardinality of a final dialgebra is unbounded.

<sup>2</sup> As a limit case, the final dialgebra still exists when  $B$  preserves the terminal object [4].  $B$  preserving the terminal object makes the category of dialgebras not very interesting, as the final dialgebra has just one element, thus all the elements of any system are bisimilar.

Final semantics is a well-established way to define behavioural equivalence of systems. Thus it may seem impossible to reconcile the lack of a final object with a well defined semantics. However, a final coalgebra provides much more than behavioural equivalence: it allows one to *compare* different systems. Such a feature is typically not exploited in the existing literature. In the absence of a final object in  $\text{Dialg}(\mathbf{F}, \mathbf{B})$ , we can still define behavioural equivalence by reasoning in terms of quotients of a system. This motivates us to study *bisimilarity quotients*. First, recall that a *quotient* of an object  $X$  in a category is the canonical representative of an equivalence class of epimorphisms from  $X$ , under the equivalence relation  $f : X \rightarrow Y \equiv g : X \rightarrow Z$  iff. there is an isomorphism  $i : Y \rightarrow Z$  such that  $i \circ f = g$ . In  $\mathbf{Set}$ , the quotients of an object form a set.

**Definition 6.** *The bisimilarity quotient of a dialgebra  $(X, f)$  is the wide pushout  $(Q, q)$  (if it exists) of the cone of quotients of  $(X, f)$  in  $\text{Dialg}(\mathbf{F}, \mathbf{B})$ . We call the diagonal  $z : (X, f) \rightarrow (Q, q)$  the canonical map of  $(X, f)$ .*

**Proposition 2.** *Let  $(Q, q)$  be the bisimilarity quotient of  $(X, f)$  and  $z$  the canonical map. For all  $x, y \in X$ ,  $x$  is bisimilar to  $y$  if and only if  $z(x) = z(y)$ . Therefore, when the bisimilarity quotient exists, bisimilarity is an equivalence relation.*

Whenever a bisimilarity quotient exists, the canonical map can be considered the semantics of a system. Notice that  $z$  is canonical, but not necessarily the unique morphism from  $(X, f)$  to  $(Q, q)$ : bisimilarity classes may be interchangeable in dialgebras.

*Example 2.* Consider the dialgebra  $g$  defined in Example 1; the dialgebra has no non-trivial quotients, so it coincides with its bisimilarity quotient. However all the isomorphisms of the carrier set are dialgebra homomorphisms. The bisimilarity classes are the same, but for the identity of the sole element of each class.

For a different example, in the semantics of a symmetric process calculus such as the *pure* variant of CCS, the bisimilarity classes of an element, and of the element obtained by replacing all the input or output actions in it with the complementary ones, can be interchanged.

Clearly, when a final object exists, the epi-mono factorisation of the final morphism yields the bisimilarity quotient. A bisimilarity quotient may exist also in the absence of a final object. Here is a sufficient condition.

**Proposition 3.** *When  $\mathbf{F}$  preserves wide (small) pushouts of epimorphisms, that is, colimits of an arbitrary small cone of epis, the bisimilarity quotient exists.*

The dialgebraic semantics of CCS in §5 is an example where the bisimilarity quotient exists, but there is no final dialgebra (by Example 1).

## 4 Interaction and observation

In this section we introduce the functors  $\mathbf{F}$  and  $\mathbf{B}$  for the dialgebraic semantics of CCS.

**Definition 7.** *The interaction and observation functors are defined as  $F(X) = X + (X \times X)$  and  $B(X) = \mathcal{P}(X)$ , respectively.*

As elements of  $X$  and  $X \times X$  are syntactically disjoint we shall denote elements of  $F(X)$  just by the elements of  $X$  and  $X \times X$ , without resorting to labels for the coproduct. We write  $(x, y) \rightarrow z$  as a shorthand for  $z \in f(x, y)$ .

An element of  $FX$  is either  $x \in X$ , representing an experiment about a process in isolation, or  $(x, y) \in X \times X$ , an experiment where two processes are allowed to interact. Elements of  $B$  are sets of processes, that are the possible non-deterministic outcomes of an experiment. We state the relevant properties of  $F$  and  $B$  in the following proposition.

**Proposition 4.** *The functors  $F$  and  $B$  preserve monos from the empty set.  $F$  preserves wide pushouts of epis. Therefore  $\text{Dialg}(F, B)$  has epi-mono factorisations, and each dialgebra in the category has a bisimilarity quotient.*

We can characterise bisimilarity in  $\text{Dialg}(F, B)$  in a similar way to LTSs.

**Proposition 5.** *The kernel of the canonical morphism into the bisimilarity quotient of an  $(F, B)$ -dialgebra  $(X, f)$  is the greatest symmetric relation  $\mathcal{R}$  such that, if  $(x, y) \in \mathcal{R}$ , for all  $w, z \in X$ ,*

- *whenever  $x \rightarrow z$ , there is  $y'$  such that  $y \rightarrow z'$  and  $(x', y') \in \mathcal{R}$ ;*
- *whenever  $(x, w) \rightarrow z$ , there is  $z'$  such that  $(y, w) \rightarrow z'$  and  $(z, z') \in \mathcal{R}$ ;*
- *whenever  $(w, x) \rightarrow z$ , there is  $z'$  such that  $(w, y) \rightarrow z'$  and  $(z, z') \in \mathcal{R}$ .*

Notice the similarity between the above proposition and Definition 1. Using dialgebras, the usual back-and-forth conditions for bisimilarity also take into account experiments; this improves over bisimilarity of reactive systems (corresponding to just using the first condition in Proposition 5) which is typically not informative enough to be useful (bisimilarity is too coarse).

## 5 The dialgebraic semantics of CCS

In §2, we have seen how to describe the semantics of CCS using a labelled transition system. Now we describe it as an  $(F, B)$ -dialgebra for the functors of Definition 7.

**Definition 8.** *Let  $X$  be the set of CCS processes. The dialgebra  $(X, f)$  for CCS is the least function obeying to the rules in Figure 3.*

We briefly comment on the rules. The first group of rules deals with processes in isolation: rule *(tau)* permits internal computation actions to be executed; Rule *(res)* allows a process in the scope of a restriction to progress; Rule *(par<sub>1</sub>)* allows one component in a parallel composition to progress independently from the others; Rule *(int)* allows any possible interaction between two processes to also happen between internal components of a process in isolation.

The second group of rules defines the semantics of interaction. Rule *(hid)* permits interaction between a process  $P$  in the scope of a restriction, and any



$$\begin{array}{c}
\frac{true}{\tau.P + Q \rightarrow P} (tau) \quad \frac{P \rightarrow P'}{(\nu a)P \rightarrow (\nu a)P'} (res) \quad \frac{P \rightarrow P'}{P \parallel Q \rightarrow P' \parallel Q} (par_1) \quad \frac{(P, Q) \rightarrow R}{P \parallel Q \rightarrow R} (int) \\
\hline
\frac{(P, Q) \rightarrow R \quad a \notin fn(Q)}{(\nu a)P, Q \rightarrow (\nu a)R} (hid) \quad \frac{(P, Q) \rightarrow R}{(P \parallel S, Q) \rightarrow S \parallel R} (par_2) \quad \frac{true}{(\bar{a}.P + S, a.Q + T) \rightarrow P \parallel Q} (syn) \\
\hline
\frac{(P, Q) \rightarrow R}{(Q, P) \rightarrow R} (sym) \quad \frac{P \rightarrow R \quad P \equiv Q, R \equiv S}{Q \rightarrow S} (str_1) \quad \frac{(P, Q) \rightarrow R \quad P \equiv S, Q \equiv T, R \equiv U}{(S, T) \rightarrow U} (str_2)
\end{array}$$

**Fig. 3.** The dialgebra for CCS.

other process  $Q$ , provided that  $a$  is not known by  $Q$ . Recall that the restricted name  $a$  can always be  $\alpha$ -converted to one which is fresh in  $Q$ . Rule  $(par_2)$  allows parallel components of a process  $P$  to interact with  $Q$  independently from each other. Rule  $(syn)$  implements synchronisation between two processes.

Rules  $(sym)$ ,  $(str_1)$ ,  $(str_2)$  simplify the definition; alternatively, one can add variants of the other rules taking into account the effects of these three schemes.

*Remark 1.* The pure CCS that we present is a particularly simple example, which is meant to be clear enough to illustrate the framework without adding complexity. However, it is not difficult to imagine how to give semantics to other calculi, by looking at their reaction semantics. The CCS with data passing, for example, would be described by just replacing the rule  $(syn)$  with the rule  $(\bar{a}z.P + S, a(x).Q + T) \rightarrow P \parallel Q[z/x]$  having  $true$  as a premise. The  $\pi$ -calculus would be handled in the same way. However, the spectrum of behavioural equivalences for nominal calculi is wide (see [12]). More work is required to precisely recover all the existing semantics (early, late, open, asynchronous, truly concurrent, etc.). We expect that dialgebras will provide a uniform treatment for these equivalence relations, starting from the corresponding reaction rules.

## 6 Comparing dialgebras

Since we already have a well-known semantics of CCS in terms of bisimilarity of an LTS, we would like to be able to state that the semantics we define coincides with the standard one. In this section we develop a technique that permits us to compare the two.

Categories of algebras or coalgebras of different functors may be compared by mapping one category into the other by composition with an appropriate natural transformation [3]. In §6.1 we show how to generalise this technique to dialgebras (of which algebras and coalgebras are special cases). The problem has first been studied in [4]. Here we improve on it by adding an intermediate “container” functor  $\mathbb{G}$ , which is crucial for the example of CCS. In §6.2 we discuss a limitation of this method and refine the construction.

### 6.1 Comparing categories of dialgebras

Consider  $\mathbf{Set}$  endofunctors  $F, B, F', B'$ . One can specify functor  $K : \mathbf{Dialg}(F, B) \rightarrow \mathbf{Dialg}(F', B')$  using  $G : \mathbf{Set} \rightarrow \mathbf{Set}$  and two natural transformations  $\lambda : F' \rightarrow GF$  and  $\mu : GB \rightarrow B'$ , as illustrated by the diagram in Figure 4.

$$\begin{array}{ccccccc}
 X & & F'X & \xrightarrow{\lambda_X} & GFX & \xrightarrow{Gf} & GBX & \xrightarrow{\mu_X} & B'X & & FX & \xrightarrow{f} & BX \\
 \downarrow h & & \downarrow F'h & & \downarrow GFh & & \downarrow GBh & & \downarrow B'h & & \downarrow Fh & & \downarrow Bh \\
 Y & & F'Y & \xrightarrow{\lambda_Y} & GFY & \xrightarrow{Gg} & GBY & \xrightarrow{\mu_Y} & B'Y & & FY & \xrightarrow{g} & BY
 \end{array}$$

**Fig. 4.** Comparing categories of dialgebras using natural transformations between  $\mathbf{Set}$  endofunctors.

Notice that dialgebras come equipped with the “underlying set” or “forgetful” functor  $U_{F,B} : \mathbf{Dialg}(F, B) \rightarrow \mathbf{Set}$  defined as  $U_{F,B}(X, f) = X$ ,  $U_{F,B}(h : (X, f) \rightarrow (Y, g)) = h$ ; this allows us to state that  $K$  is *concrete*.

**Theorem 1.** *Two natural transformations  $\lambda : F' \rightarrow GF$  and  $\mu : GB \rightarrow B'$  determine a functor  $K : \mathbf{Dialg}(F, B) \rightarrow \mathbf{Dialg}(F', B')$  as  $K(X, f) = (X, \mu_X \circ Gf \circ \lambda_X)$ ,  $K(h : (X, f) \rightarrow (Y, g)) = h$ .  $K$  is concrete, that is:  $U_{F',B'} \circ K = U_{F,B}$ .*

*Proof.* For all dialgebra morphisms  $h$ , the diagram in Fig 4 commutes. The middle square commutes since  $h$  is a dialgebra homomorphism and  $G$  is a functor. The left and right ones by naturality of  $\lambda$  and  $\mu$ . Therefore  $h$  is a dialgebra homomorphism from  $K(X, f)$  to  $K(Y, g)$ .  $K$  is concrete by definition and easily checked to be a functor.

As a consequence of  $K$  being concrete, we have the following corollary.

**Corollary 1.** *If  $x, y \in X$  are bisimilar in  $(X, f)$  then they are so in  $K(X, f)$ .*

### 6.2 Comparing dialgebras

The framework of Sec. 6.1 is more restrictive than necessary. Observe that both  $\lambda$  and  $\mu$  have to be defined for each set  $X$ . But when comparing say, two different semantics of CCS, we are only interested in the two dialgebras, and in the objects that may be reached from them by a dialgebra epimorphism. By only considering this subclass of objects, we are allowed in the definition of  $\lambda$  and  $\mu$  to use specific features of a given dialgebra (e.g. we may use fact that the underlying set  $X$  is the carrier of an algebra, or refer to specific elements of  $X$ ).

Although proofs in this section require quite a bit of categorical reasoning, the results mostly depend on the choice of an appropriate *bisimulation invariant*

(Definition 11), which is not a difficult task by itself and encapsulates the complexity of the framework in a simple definition. We support this claim with §7, where we show how our main theorem is applied to compare the dialgebraic and LTS semantics of CCS. The choice of invariants is driven by a simple intuition on how side effects and interactions are related in CCS. Furthermore, the added complexity is only relevant to compare different semantics, and does not directly affect usage of dialgebras as described in the previous sections.

Consider the problem of defining a coalgebra  $(X, f')$  for the functor of §2 out of the dialgebra for CCS of §5. We can employ “witness processes” such as  $a.0$  in experiments such as  $(x, a.0)$ . For each  $x' \in f(x, a.0)$ , we let  $f'(x)$  contain the labelled transition  $(\bar{a}, x')$ . The idea sounds promising, but in the process we need to refer to specific elements of  $X$ , namely the witness processes. Natural transformations between **Set** endofunctors are not allowed to depend on a specific set.

However, our comparison technique is not defeated by this fact. What we need is to restrict our attention to natural transformations between functors whose codomain is **Set**, but whose domain is not. The framework may look complicated, but by Theorem 2, such natural transformations are specified by single functions which we call *bisimulation invariant*, serving as an adaptation layer between dialgebras of different type. §7 shows that in practice the definition of these functions, and the proofs of their invariance, may be very simple.

Let **E** be the subcategory of epimorphisms of  $\mathbf{Dialg}(\mathbf{F}, \mathbf{B})$ . Given a dialgebra  $(X, f)$ , consider the coslice category  $(X, f)/\mathbf{E}$ . Objects of the category are arrows in **E** whose domain is  $(X, f)$ . Arrows of the category are commuting morphisms of **E**. In other words the category is the preorder of epimorphisms from  $(X, f)$ .

**Definition 9.** *We assume a chosen full subcategory  $\mathbf{R}_{(X, f)}$ , of  $(X, f)/\mathbf{E}$ , having the following properties:*

- *for each object  $h$  of  $(X, f)/\mathbf{E}$  there is at least one object  $h'$  of  $\mathbf{R}_{(X, f)}$  with a commuting arrow  $k : h \rightarrow h'$ ;*
- *the identity  $id_{(X, f)}$  is an object.*

The first condition says that  $\mathbf{R}_{(X, f)}$  contains enough epimorphisms to characterise bisimilarity: whenever  $x$  and  $y$  are identified by a morphism  $h$ , there is a morphism  $h'$  that identifies them and belongs to the category. The second condition is necessary for our definitions. The purpose of  $\mathbf{R}_{(X, f)}$  is to serve as a domain for functors into **Set**, so that natural transformations between them may have a much more specific definition. Not only this allows one to define a natural transformation making explicit reference to  $(X, f)$ , its elements and its properties, but also one can define a separate map for each dialgebra morphism, in a subcategory which is sufficiently rich to characterise bisimilarity. This restricted setting is sufficient to compare different kinds of dialgebras, as we shall see. The fact that  $\mathbf{R}_{(X, f)}$  may be chosen as a subcategory of  $(X, f)/\mathbf{E}$  allow us to restrict the domain of our functors even further. For example, for one side of the comparison between the two semantics of CCS in §7, we chose the morphisms in  $\mathbf{R}_{(X, f)}$  so that their kernels are congruences with respect to the parallel operator.

However, endofunctors from **Set** to **Set** need to be replaced with functors from  $\mathbf{R}_{(X,f)}$  to **Set**. The following definition is used for the purpose.

**Definition 10.** For each functor  $F : \mathbf{Set} \rightarrow \mathbf{Set}$ , define its lifting  $\bar{F} : \mathbf{R}_{X,f} \rightarrow \mathbf{Set}$  as  $\bar{F} = F \circ U_{F,B} \circ \text{cod}$ , where  $\text{cod} : (X,f)/E \rightarrow E$  is the codomain functor, mapping objects (arrows in  $\text{Dialg}(F,B)$ ) to their codomains, and arrows to themselves.

The functor  $\bar{F}$  acts on objects as  $\bar{F}(p : (X,f) \rightarrow (Y,g)) = F(Y)$  and on arrows as  $\bar{F}(k) = k$ .

Next, we prove that natural transformations indexed by  $\mathbf{R}_{(X,f)}$  may be specified by single functions, obeying to a condition that we call *bisimulation invariance*. Notice that, since  $\mathbf{R}_{(X,f)}$  is a full subcategory containing the identity of a coslice category, each arrow  $h : (X,f) \rightarrow (Y,g)$  can be regarded as both an object of  $\mathbf{R}_{X,f}$  and an arrow in the same category from  $\text{id}_{(X,f)}$  to  $h$  itself. In the following, we refer to the arrow as  $\hat{h}$  to avoid confusion.

**Definition 11.** Given two functors  $F, G : \mathbf{R}_{X,f} \rightarrow \mathbf{Set}$ , consider a function  $k : F(\text{id}_{(X,f)}) \rightarrow G(\text{id}_{(X,f)})$ . Call  $k$  bisimulation invariant with respect to  $(X,f)$  and  $\mathbf{R}_{(X,f)}$  from  $F$  to  $G$  iff. for all  $x_1, x_2 \in X$ , and for each arrow  $\hat{h} : \text{id}_{(X,f)} \rightarrow h$  in  $\mathbf{R}_{X,f}$ , we have  $F\hat{h}(x_1) = F\hat{h}(x_2) \implies G\hat{h}(k(x_1)) = G\hat{h}(k(x_2))$ .

In the following we call  $k$  simply *invariant* when  $(X,f)$ ,  $\mathbf{R}_{(X,f)}$ ,  $F$  and  $G$  are clear from the context. Such a property of a function may seem difficult to prove; however, it is actually easier than proving naturality. Commutativity is required only for a given class of morphisms; these are also guaranteed to preserve and reflect bisimilarity.

We use invariance in the following proposition, which depends on  $F$  preserving epis. This is true for the lifting  $\bar{F}$  of a **Set** endofunctor, as all **Set** endofunctors preserve epis, and so do the  $\text{cod}$  and  $U$  functors used in Definition 10.

**Theorem 2.** Consider two functors  $F, G : \mathbf{R}_{X,f} \rightarrow \mathbf{Set}$ , with  $F$  preserving epis. There is a one-to-one correspondence between natural transformations  $\delta : F \rightarrow G$  and invariants from  $F$  to  $G$ . Each natural transformation  $\delta$  is uniquely determined by  $\delta_{\text{id}_{(X,f)}}$ , which is invariant; conversely, for each invariant  $k$  there is a unique natural transformation  $\delta$  such that  $\delta_{\text{id}_{(X,f)}} = k$ .

*Proof.* Consider any object  $h : (X,f) \rightarrow (Y,g)$  of  $\mathbf{R}_{X,f}$ . We show that  $\delta_h$  is uniquely determined by  $\delta_{\text{id}_{(X,f)}}$ . Consider the commuting square

$$\begin{array}{ccc} F(\text{id}_{(X,f)}) & \xrightarrow{\delta_{\text{id}_{(X,f)}}} & G(\text{id}_{(X,f)}) \\ \downarrow F\hat{h} & & \downarrow G\hat{h} \\ Fh & \xrightarrow{\delta_h} & Gh \end{array}$$

Since  $\hat{h}$  is epic, and  $F$  preserves epis, a commuting  $\delta_h$  is uniquely determined by  $\delta_{id_{(X,f)}}$ : by commutativity, for all  $x \in F(id_{(X,f)})$ , we have  $\delta_h(F\hat{h}(x)) = G\hat{h}(\delta_{id_{(X,f)}}(x))$ . If  $id_{(X,f)}$  is invariant, this equation defines a function  $\delta_h$ . Conversely, if  $\delta$  is natural, then  $\delta_{id_{(X,f)}}$  is invariant by definition.

We can restate Theorem 1 in terms of natural transformations  $\lambda, \mu$  between functors from  $\mathbf{R}_f$  to  $\mathbf{Set}$ ; this is described by the diagram in Figure 5.

$$\begin{array}{ccccccc}
\bar{F}'(id) = F'X & \xrightarrow{\lambda_{id}} & G\bar{F}(id) = GFX & \xrightarrow{Gf} & G\bar{B}(id) = GBX & \xrightarrow{\mu_{id}} & \bar{B}'(id) = B'X \\
\downarrow \bar{F}'h = F'h & & \downarrow G\bar{F}h = GFh & & \downarrow G\bar{B}h = GBh & & \downarrow \bar{B}'h = B'h \\
\bar{F}'(h) = F'Y & \xrightarrow{\lambda_h} & G\bar{F}(h) = GFY & \xrightarrow{Gg} & G\bar{B}(h) = GBY & \xrightarrow{\mu_h} & \bar{B}'(h) = B'Y
\end{array}$$

**Fig. 5.** Comparing dialgebras using natural transformations between functors from  $\mathbf{R}_{X,f}$  to  $\mathbf{Set}$ . Here  $id$  is  $id_{(X,f)}$ .

**Theorem 3.** *Given an  $(F, B)$ -dialgebra  $(X, f)$ , a category  $\mathbf{R}_{(X,f)}$  as in Definition 11, a functor  $G$ , and two invariants  $\lambda$  from  $F'$  to  $G\bar{F}$ ,  $\mu$  from  $G\bar{B}$  to  $\bar{B}'$ , consider the  $(F', B')$ -dialgebra  $(X, f^{\lambda, \mu})$  where  $f^{\lambda, \mu} = \mu \circ Gf \circ \lambda$ . Whenever two elements are bisimilar in  $(X, f)$ , then they are bisimilar in  $(X, f^{\lambda, \mu})$ .*

*Proof.* Since  $\lambda$  and  $\mu$  are invariant, they determine corresponding natural transformations by Proposition 2. If two elements  $x_1$  and  $x_2$  are bisimilar in  $(X, f)$ , then there is a morphism of  $(F, B)$ -dialgebras  $h : (X, f) \rightarrow (Y, g)$  such that  $h(x_1) = h(x_2)$ . By epi-mono factorisations (Proposition 1), we can assume w.l.o.g. that  $h$  is epic, and by the condition in Definition 9 we can also assume that it is an object of  $\mathbf{R}_{(X,f)}$  (formally one sees that there is an object in the subcategory that identifies  $x_1$  and  $x_2$ ). Let  $g' = \mu_h \circ Gg \circ \lambda_h$ . Notice that  $(Y, g')$  is an  $(F', B')$ -dialgebra. Consider the diagram in Figure 5 and the equations therein. These equations hold by Definition 10. The diagram commutes: the middle square since  $h$  is a dialgebra homomorphism and  $G$  a functor, the left and right ones by naturality of  $\lambda$  and  $\mu$ . From commutativity of the perimeter, we see that  $h$  is also a morphism of  $(F', B')$ -dialgebras from  $(X, f^{\lambda, \mu})$  to  $(Y, g')$ , therefore  $x_1$  and  $x_2$  are bisimilar in  $(X, f^{\lambda, \mu})$ .

## 7 Comparing the dialgebra and coalgebra for CCS

In this section, we use Theorem 3 to compare the semantics for CCS from Definition 8 to bisimilarity in the well-known labelled transition system.

The dialgebra describes processes as they interact, with no explicit notion of side effect. The coalgebra describes processes in isolation, and their side effects. It is not difficult to imagine how the two kinds of semantics can be compared. In one direction, starting from the coalgebra, we may define a  $(F, B)$ -dialgebra on the same carrier; for processes in isolation, we run one step of the LTS, and then turn all the  $\tau$  transitions into observed results; for interaction between pairs of elements, we run one step of the LTS on each element, and let interaction happen whenever an input (or an output) is matched by the complementing action. In the other direction, starting from the dialgebra, we may define a T-coalgebra, by letting  $\tau$  transitions correspond to the observations that are made on a process in isolation, and by running experiments in which we let a process and a “witness process” such as  $a.\emptyset$  interact. The results of the experiments are labelled with a corresponding action, e.g. in our example we would use the label  $\bar{a}$ .

In the rest of the section we will closely implement the above plan. From now on, we let  $(X, f)$ ,  $(X, g)$ , and the functors  $F, B$  be the dialgebra and the coalgebra for CCS, and the functors from §4; we let  $T(X) = \mathcal{P}(L \times X)$ .

**Definition 12.** We define the functions  $\delta : \mathbf{FTX} \rightarrow \mathbf{BX}$ ,  $\lambda : X \rightarrow \mathbf{TFX}$ ,  $\mu : \mathbf{TBX} \rightarrow \mathbf{TX}$  as follows:

$$\delta(e) = \begin{cases} \{x \mid (\tau, x) \in p\} & \text{if } e = p \in \mathbf{TX} \\ \{x \parallel y \mid \exists a \in \mathcal{C}. ((a, x) \in p_1 \wedge (\bar{a}, y) \in p_2) \vee \\ \quad \vee ((\bar{a}, x) \in p_1 \wedge (a, y) \in p_2)\} & \text{if } e = (p_1, p_2) \in \mathbf{TX} \times \mathbf{TX} \end{cases}$$

$$\lambda(x) = \{(\tau, x)\} \cup \{(a, (x, \bar{a}.\emptyset)) \mid a \in \mathcal{C}\} \cup \{(\bar{a}, (x, a.\emptyset)) \mid a \in \mathcal{C}\}$$

$$\mu(q) = \{(l, x) \mid \exists q'. (l, q') \in q \wedge x \in q'\}$$

Notice how the definition of  $\delta$  uses the fact that  $X$  is also the carrier of the initial algebra, therefore the parallel composition  $x \parallel y$  is defined. An appropriate choice of  $R_{(X, g)}$  makes  $\delta$  an invariant. Also, the definition of  $\lambda$  uses specific elements of  $X$ , such as  $\bar{a}.\emptyset$ . On the other hand,  $\mu$  is independent of  $X$  and extends to a natural transformation from  $\mathbf{TB}$  to  $\mathbf{T}$ .

**Proposition 6.** Let  $E$  be the subcategory of  $\mathbf{Dialg}(F, B)$  of epis whose domain is  $(X, f)$ , and  $E'$  the subcategory of  $\mathbf{Dialg}(\mathbf{Id}, \mathbf{T})$  of epis whose domain is  $(X, g)$ . Let  $R_{(X, f)}$  be the coslice  $(X, f)/E$ , and  $R_{(X, g)}$  be the full subcategory of  $(X, g)/E'$  whose objects  $h$  commute with the parallel operator, that is,  $h(x) = h(x') \wedge h(y) = h(y') \implies h(x \parallel y) = h(x' \parallel y')$ . Then:

- $id_{\mathbf{FX}}$  is invariant for  $(X, g)$  and  $R_{(X, g)}$  from  $\bar{\mathbf{F}}$  to  $\mathbf{F} \bar{\mathbf{Id}} = \bar{\mathbf{F}}$ ;
- $\delta$  is invariant for  $(X, g)$  and  $R_{(X, g)}$  from  $\mathbf{F} \bar{\mathbf{T}}$  to  $\mathbf{B}$ ;
- $\lambda$  is invariant for  $(X, f)$  and  $R_{(X, f)}$  from  $\bar{\mathbf{Id}}$  to  $\mathbf{T} \bar{\mathbf{F}}$ ;
- $\mu$  is invariant for  $(X, f)$  and  $R_{(X, f)}$  from  $\mathbf{T} \bar{\mathbf{B}}$  to  $\bar{\mathbf{T}}$ .

In Proposition 6, we are allowed to let  $R_{(X,g)}$  only contain those homomorphisms that commute with the parallel operator, therefore strengthening the hypothesis for invariance, which facilitates the proof.

We can use Theorem 3, twice. Let  $G = F$ ; we obtain the  $(F, B)$ -dialgebra  $(X, g^{id_{FX}, \delta})$  where  $f^{id_{FX}, \delta} = \delta \circ Fg$ . Similarly, let  $G = T$ ; then we derive the  $(Id, T)$ -dialgebra, that is,  $T$ -coalgebra,  $(X, f^{\lambda, \mu})$  with  $f^{\lambda, \mu} = \mu \circ Tf \circ \lambda$ .

So far, we have mapped the dialgebra of CCS into a coalgebra, and the coalgebra into an  $(F, B)$ -dialgebra. However, no link is established between  $f$  and  $g^{id, \delta}$ , or  $g$  and  $f^{\lambda, \mu}$ . We conclude the paper by proving coincidence of the two semantics. For this, we need the following lemma.

**Lemma 1.** *For all channels  $a$  and elements  $x, y, z$ , the following holds:*

$$\begin{aligned}
- & x \xrightarrow{\tau} y \iff x \rightarrow y; \\
- & x \xrightarrow{a} y \iff (x, \bar{a}.\emptyset) \rightarrow y; \\
- & x \xrightarrow{\bar{a}} y \iff (x, a.\emptyset) \rightarrow y; \\
- & (x, y) \rightarrow z \iff \exists b, x', y'. z \equiv x' \parallel y' \wedge \\
& \quad \wedge ((x \xrightarrow{b} x' \wedge y \xrightarrow{\bar{b}} y') \vee (x \xrightarrow{\bar{b}} x' \wedge y \xrightarrow{b} y')).
\end{aligned}$$

**Proposition 7.** *Let  $(X, f)$  and  $(Y, g)$  be the dialgebra and the coalgebra for CCS. We have that  $f = g^{id_{FX}, \delta}$  and  $g = f^{\lambda, \mu}$ . Therefore, bisimilarity in  $(X, f)$  and in  $(X, g)$  is the same.*

## 8 Conclusions and future work

Dialgebras have distinctive features with respect to coalgebras. The most important one is that there is no final dialgebra, therefore no universal model. This forces one to reason in terms of quotients. The locality which is intrinsic to the definition of a dialgebra deserves in our opinion a more thorough investigation.

One problem of the dialgebraic approach is apparent in Definition 8: the rules may look odd with two versions of the parallel and restriction operators. Structural rules (the last row in the definition) seem somewhat redundant, too. We foresee that this issue will be address by answering the fundamental question of how to axiomatise dialgebras. Future research should identify equational or modal formalisms that uniquely identify dialgebraic structures starting from a smaller set of rules or schemes. In this respect, and for compositionality, we will investigate a theory of dialgebras, that are inductively defined on a term algebra. The interplay between adequate logics for dialgebras, and equational logic on terms, may lead to new insights on algebraic and coalgebraic specifications.

Another open problem is the implementation and verification of dialgebras. Coalgebras have an associated partition refinement procedure that computes the bisimilarity quotient of a system, by the means of iteration along the terminal sequence of the functor  $T$ . Promising work in progress has been devoted to finding a generalisation of this procedure to compute the bisimilarity quotient of a dialgebra, and will appear in future work.

## References

1. Hagino, T.: A Categorical Programming Language. PhD thesis, University of Edinburgh (1987)
2. Poll, E., Zwanenburg, J.: From algebras and coalgebras to dialgebras. *Electronic Notes in Theoretical Computer Science* **44**(1) (2001) 289 – 307
3. Rutten, J.J.M.M.: Universal coalgebra: a theory of systems. *Theoretical Computer Science* **249**(1) (2000) 3 – 80
4. Voutsadakis, G.: Universal dialgebra: unifying algebra and coalgebra. *Far East Journal of Mathematical Sciences* **44**(1) (2010) 1–53
5. Ciancia, V.: Interaction and observation, categorically. In: *Proceedings Fourth Interaction and Concurrency Experience*. Volume 59. (2011) 25–36
6. Leifer, J.J., Milner, R.: Deriving bisimulation congruences for reactive systems. In: *In Proc. of CONCUR 2000*, 2000. LNCS 1877, Springer (2000) 243–258
7. Sewell, P.: From rewrite rules to bisimulation congruences. *Theor. Comput. Sci.* **274** (March 2002) 183–230
8. Sassone, V., Sobocinski, P.: Deriving bisimulation congruences using 2-categories. *Nord. J. Comput.* **10**(2) (2003) 163–
9. Turi, D., Plotkin, G.: Towards a mathematical operational semantics. In: *12th Annual IEEE Symposium on Logic in Computer Science (LICS)*, IEEE Computer Society (1997) 280–291
10. Milner, R.: *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc. (1982)
11. Staton, S.: Relating coalgebraic notions of bisimulation: with applications to name-passing process calculi. In: *3rd international conference on Algebra and coalgebra in computer science*, Springer-Verlag (2009) 191–205
12. Sangiorgi, D., Walker, D.: *The Pi-Calculus - a theory of mobile processes*. Cambridge University Press (2001)

## A Proofs

*Proof.* (Proposition 1) Consider a dialgebra homomorphism  $h : (X, f) \rightarrow (Y, g)$  and the diagram of Figure 2. First notice that the epimorphisms in  $Dialg(\mathbf{F}, \mathbf{B})$  are just the epimorphisms in  $\mathbf{Set}$  that are dialgebra morphisms. Since  $\mathbf{Set}$  has epi-mono factorisations, we can factor every arrow  $j$  in the diagram, including  $h$  itself, as  $j_m \circ j_e$  where  $j_e$  is epic, and  $j_m$  is monic. Call  $X'$  the codomain of  $h_e$ . We are going to endow  $X'$  with a dialgebra structure  $(X', f')$  so that  $h_e$  is a dialgebra morphism from  $(X, f)$  to  $(X', f')$ .

Since all set functors preserve all epis, and all monos except possibly those with an empty domain, assume that either  $\mathbf{F}$  and  $\mathbf{B}$  preserve the monos with an empty domain, or that  $X \neq \emptyset$ . In the latter case,  $X' \neq \emptyset$  as there are no arrows whose codomain is the empty set. Thus,  $Fh_e$  epic, and  $Fh_m$  monic. Then by uniqueness of epi-mono factorisations,  $Fh = F(h_m \circ h_e) = Fh_m \circ Fh_e$  is the epi-mono factorisation of  $Fh$ . Similarly  $Bh = Bh_m \circ Bh_e$  is the unique factorisation of  $Bh$ . Consider the pushout  $P$  of  $f_e, Fh_e$ , and the injection  $f'_e : FX' \rightarrow P$ . There is a unique commuting arrow  $p : P \rightarrow BY$ . Call  $p' : FX \rightarrow P$  the diagonal of the pushout. All the arrows we mentioned except  $p$  are epi. Similarly, consider the pullback  $Q$  of  $g_m, Bh_m$ . There is a unique commuting arrow  $q : FX \rightarrow Q$ . Also consider the projection  $g'_m : Q \rightarrow BX'$  of the pullback, and the diagonal  $q' : Q \rightarrow BX$ . All the arrows in this sub-diagram except  $q$  are mono. We have



two morphisms  $r = p \circ p'$  and  $s = q' \circ q$  that commute with the outer square, therefore they are equal. Now factor  $p = p_m \circ p_e$  and  $q = q_m \circ q_e$ . Then  $r = p_m \circ p_e \circ p'$  with  $p_e \circ p'$  epi. Also  $s = q' \circ q_m \circ q_e$  with  $q' \circ q_m$  mono. Therefore we have two epi-mono factorisations of the arrow  $r = s$ , thus there is an isomorphism  $i : \text{cod}(p_e) \rightarrow \text{dom}(q_m)$ . The composite  $f' = g'_m \circ q_m \circ i \circ p_e \circ f'_e$  is an arrow from  $\mathbf{F}X'$  to  $\mathbf{G}X'$ . It is easy to see that  $h_e$  is a dialgebra homomorphism from  $(X, f)$  to  $(X', f')$ . Commutativity is by diagram chasing, and uniqueness by  $\mathbf{F}h_e$  epi.

*Proof.* (Proposition 2) If  $k(x) = k(y)$  then  $x$  and  $y$  are bisimilar by definition. Conversely, given a homomorphism  $h$  such that  $h(x) = h(y)$ , by epi-mono factorisations we can assume  $h$  is epic ( $X$  is certainly non-empty since we assume an element  $x$ ); w.l.o.g. assume  $h$  is a quotient (as bisimilarity is the same in a class of isomorphic epis). Thus, there is an epi  $h'$  such that  $h' \circ h = k$ , thus  $k(x) = k(y)$ . As bisimilarity is the kernel of a morphism, it is an equivalence.

*Proof.* (Proposition 3) Call  $P$  the cocone of epimorphisms from  $(X, f)$ . First, observe that  $P$  is a small diagram, that is, its objects and morphisms form sets. This is since the cone of  $X$  in  $\mathbf{Set}$  forms a set  $S$ , and  $P$  can be at most as large as the product  $S \times \mathbf{B}X^{\mathbf{F}X}$  (the possible quotients are as many as the dialgebras whose carrier is a quotient of  $X$  in  $\mathbf{Set}$ ). Since  $\mathbf{Set}$  is cocomplete, the pushout  $Q$  of the morphisms in  $P$  exists in  $\mathbf{Set}$ . It is not difficult to see that the category of  $(\mathbf{F}, \mathbf{B})$ -dialgebras has all colimits that the base category ( $\mathbf{Set}$  in our case) has and  $\mathbf{F}$  preserves (see Theorem 14 in [4]). Therefore if  $\mathbf{F}$  preserves wide pushouts, there is a pushout  $(Q, q)$  of  $P$  as a diagram in  $\mathbf{Dialg}(\mathbf{F}, \mathbf{B})$ . The carrier is  $Q$ ; the dialgebra map  $q$  is the unique commuting morphism from  $\mathbf{F}Q$ , which is a pushout of all the  $\mathbf{F}h$  for  $h$  in  $P$ , to  $\mathbf{B}Q$ , that forms a commuting diagram with all the dialgebras that are codomains of morphisms in  $P$ .

*Proof.* (Proposition 4)  $\mathbf{F}$  preserves the initial object and  $\mathbf{B}$  sends it into 1. Therefore they both preserve the specified monos. By relying on the fact that all epi split, it is easy to see that  $\mathbf{F}$  preserves binary pushouts of epis, even though it fails to preserve pushouts in general. Since  $\mathbf{F}$  also preserves filtered colimits, it preserves wide pushouts of epis.

*Proof.* (Proposition 5) We omit this proof, which is patterned after the proof that LTS bisimilarity coincides with the relation induced by the kernel of morphisms in the corresponding category of coalgebras. A very similar proof for dialgebras can be also found in [5], Theorem 1.

*Proof.* (Proposition 6) Note that  $\mathbf{R}_{(X, g)}$  respects the conditions in Definition 9 because both the required identity and the final coalgebra are included in it. The result on the identity function being invariant is obvious.

For  $\delta$ , suppose  $\mathbf{F}Th(e_1) = \mathbf{F}Th(e_2)$ . By definition of  $\mathbf{F}$ , this implies either  $e_1 = p, e_2 = p'$  or  $e_1 = (p_1, p_2), e_2 = (p'_1, p'_2)$ , where  $p, p', p_1, p_2, p'_1, p'_2 \in \mathbf{T}X$ .

When  $e_1 = p, e_2 = p'$ , we have  $Th(p) = Th(p')$ . Thus  $\mathbf{B}h(\delta(p)) = \{h(x) \mid (\tau, x) \in p\} = \{x \mid (\tau, x) \in Th(p)\} = \{x \mid (\tau, x) \in Th(p')\} = \mathbf{B}h(\delta(p'))$ .

In the other case, we have  $\mathbf{B}h(\delta(p_1, p_2)) = \{h(x \parallel y) \mid \Gamma(p_1, p_2, x, y)\}$  where  $\Gamma$  is a shorthand for the defining condition of the set. Consider the set of pairs  $S = \{(h(x), h(y)) \mid \Gamma(p_1, p_2, x, y)\}$ . Under the hypothesis  $\mathbf{F}Th(e_1) = \mathbf{F}Th(e_2)$ , we have  $S =$

$\{(x, y) \mid \Gamma(\text{Th}(p_1), \text{Th}(p_2), x, y)\} = \{h(x), h(y) \mid \Gamma(p'_1, p'_2, x, y)\}$ . We shall prove that if this equality holds, then  $\text{Bh}(\delta(p_1, p_2)) = \text{Bh}(\delta(p'_1, p'_2))$ . The equality implies that, for all  $x, y$  such that  $\Gamma(p_1, p_2, x, y)$ , there are  $x', y'$  such that  $\Gamma(p_1, p_2, x', y')$ , and  $h(x) = h(x'), h(y) = h(y')$ . Therefore, by how we chose the objects of  $\mathbf{R}_{X,g}$ , we have  $h(x \parallel y) = h(x' \parallel y')$ .

The cases for  $\lambda$  and  $\mu$  are handled similarly.

*Proof.* (Lemma 1) The proof is a simple induction on the derivation. Notice that the rules dealing with structural congruence permit us to prove equality of the destination states; otherwise this lemma would hold only up-to structural congruence. We only show the most complicated part of the proof, which is the right to left direction of the last case. Suppose  $x \xrightarrow{b} x'$  and  $y \xrightarrow{\bar{b}} y'$ , so we have finite derivations for these transitions. We construct a finite derivation of  $(x, y) \rightarrow z$  by induction on the sum of the lengths of the two derivations. The base case is when the last rule of both derivations is *(pre)*; then we can apply Rule *(syn)* from Figure 3 and obtain the thesis. Otherwise, we look at the last step in the derivation of  $x \xrightarrow{b} x'$ . If either *(res)*, *(par)* or *(str)* from Figure 1 is used, we apply the inductive hypothesis to the transitions in the premises, obtaining a derivation of a transition in the dialgebra, and then we conclude the derivation of  $(x, y) \rightarrow z$  either by Rule *(hid)*, *(par<sub>2</sub>)* or *(str<sub>2</sub>)* from Figure 3, respectively. For *(hid)* to be applied correctly, we may assume that all bound names are sufficiently fresh, because of  $\alpha$ -conversion. When Rule *(pre)* is the last rule used to derive  $x \xrightarrow{b} x'$ , we construct the derivation of  $(x, y) \rightarrow z$  by applying Rule *(sym)*, and then by applying the inductive hypothesis (notice that the argument is symmetric in the polarity of a label (input or output)).

*Proof.* (Proposition 7) Let us look at equality of  $(X, g)$  and  $(X, f^{\lambda, \mu})$ . We have  $f(\lambda(x)) = \{(\tau, f(x))\} \cup \{(a, f(x, \bar{a}.\emptyset)) \mid a \in \mathcal{C}\} \cup \{(\bar{a}, f(x, a.\emptyset)) \mid a \in \mathcal{C}\}$ . Then, by Lemma 1, this set is equal to  $\{x' \mid x \xrightarrow{\tau} x'\} \cup \{(a, \{x' \mid x \xrightarrow{a} x'\}) \mid a \in \mathcal{C}\} \cup \{(\bar{a}, \{x' \mid x \xrightarrow{\bar{a}} x'\}) \mid a \in \mathcal{C}\}$ . Then it is immediate that  $f^{\lambda, \mu}(x) = \mu(\text{Tf}(\lambda(x))) = g(x)$ . For equality of  $(X, f)$  and  $(X, g^{id, \delta})$ , we have  $\delta(\text{Fg}(x)) = \{x \mid x \xrightarrow{\tau} x'\}$ , which by Lemma 1 is equal to  $\{x \mid x \rightarrow x'\} = f(x)$ . The binary case is analogous, using the last case of Lemma 1.